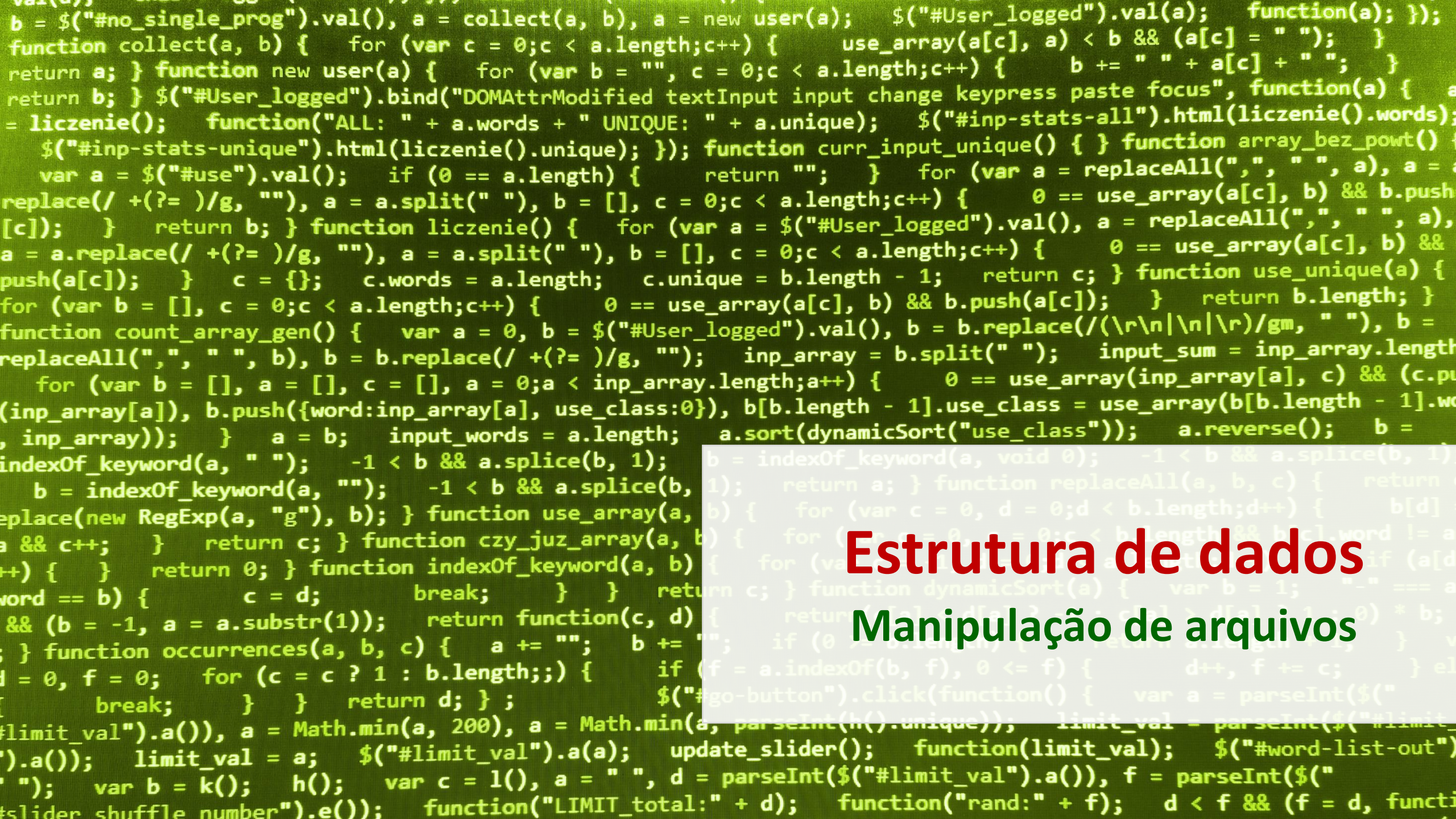
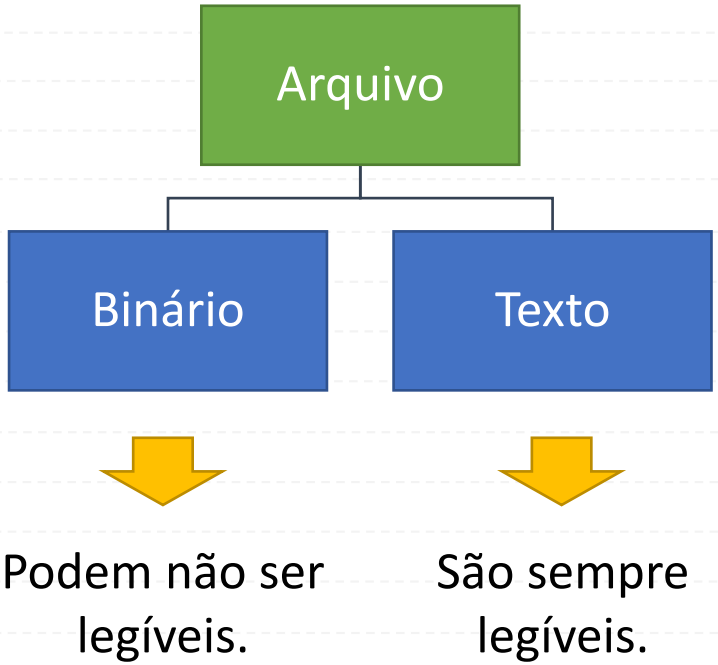

Estrutura de dados
Manipulação de arquivos

Definições

Um **arquivo** é uma coleção de bytes armazenados em um dispositivo de armazenamento.
Em C, existem dois tipos de arquivos:



Armazena dados como estão organizados na memória do computador (arquivos compactados, executáveis)

```
01001101010001011101010101010
10111010110110101010110110001
01010100010101010000101010100
01110101011001001101010001011
10101010101010111010110110101
01011011000101010100010101010
00010101010001110101011001001
10101000101110101010101010111
01011011010101011011000101010
10001010101000010101010001110
```

Nos arquivos de texto, os dados estão armazenados como sequência de caracteres, que nos são legíveis.

Este é um arquivo texto, e os dados estão armazenados (e são traduzidos) como sequência de caracteres.

Manipulando um arquivo

São três as funções iniciais para manipular arquivos (criando uma variável do tipo arquivo, abrindo o arquivo apontado pela variável do tipo arquivo e fechando o arquivo apontado pela variável do tipo arquivo).

Detalhando...

- `FILE *fopen(char *NomeArquivo, char Modo);`

Função para **abrir** um arquivo.

Exemplo:

```
arquivo = fopen("arquivo.txt", “r");
```

Manipulando um arquivo

São três as funções iniciais para manipular arquivos (criando uma variável do tipo arquivo, abrindo o arquivo apontado pela variável do tipo arquivo e fechando o arquivo apontado pela variável do tipo arquivo).

Detalhando...

- `int fclose (FILE * fp);`

Função para **fechar** um arquivo aberto.

Exemplo:

```
fclose(arquivo);
```

Definições

Observações:

(1) A linguagem C não possui funções que automaticamente leiam todas as informações de um arquivo, mas é possível manipula-los no código.

(2) A biblioteca necessário para manipular um arquivo é a <stdio.h>

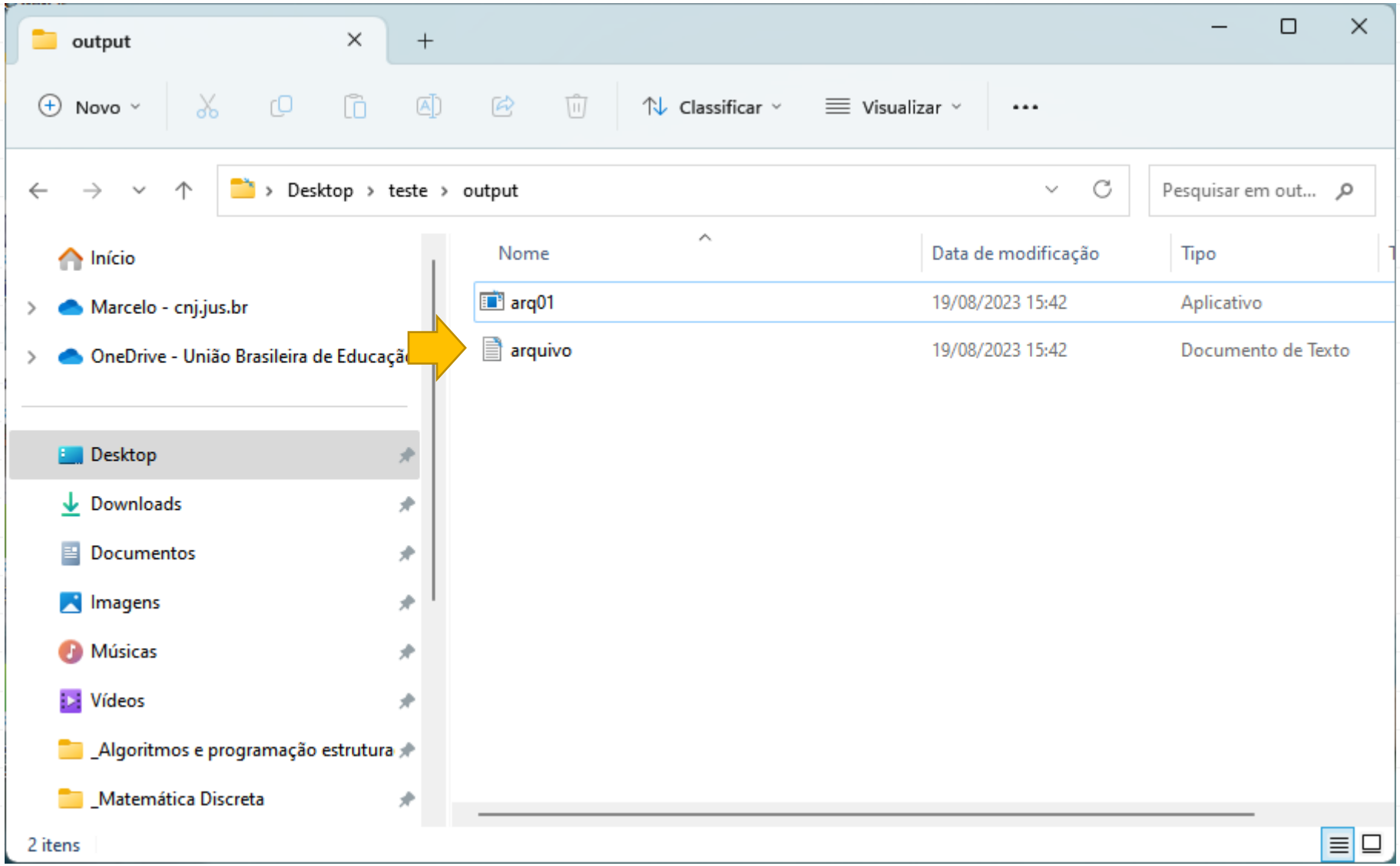
Modos de abertura

Em C, os modos de abertura de arquivo são usados para especificar como um arquivo será aberto e manipulado. Os modos de abertura são representados por strings passadas como argumentos para a função fopen(). Os modos de abertura mais comuns são:

"r" (Read)	Abre o arquivo para leitura. Se o arquivo não existir, a operação de abertura falhará.
"w" (Write)	Abre o arquivo para escrita. Se o arquivo já existir, seu conteúdo será apagado. Se o arquivo não existir, um novo arquivo será criado.
"a“ (Append)	Abre o arquivo para escrita, mas acrescenta dados ao final do arquivo, em vez de sobrescrevê-lo. Se o arquivo não existir, um novo arquivo será criado.

Q01

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main() {

    FILE *arquivo;

    arquivo = fopen("arquivo.txt", "w");

    if (arquivo == NULL) {
        printf("Erro ao abrir o arquivo.\n");
        exit(1);
    }

    fclose(arquivo);

    return 0;
}
```


output
Novo
Classificar
Visualizar
Desktop › teste › output
Pesquisar em out...
Início
Marcelo - cnj.jus.br
OneDrive - União Brasileira de Educaçã
Desktop
Downloads
Documentos
Imagens
Músicas
Vídeos
_Algoritmos e programação estrutura
_Matemática Discreta
Nome
Data de modificação
Tipo
arq01
19/08/2023 15:42
Aplicativo
arquivo
19/08/2023 15:42
Documento de Texto
2 itens

fopen()

Avaliando o valor de retorno do **fopen()**:

- Erros podem ocorrer (por exemplo: abrir para leitura um arquivo que não existe);
- Em falhas, o retorno de fopen() é indicado por **NULL**.

```
    arquivo = fopen("arquivo.txt", "wb");

    if (arquivo == NULL) {
        printf("Erro ao abrir o arquivo.\n");
        exit(1);
    }
```

exit() é uma função da biblioteca **stdlib.h** da linguagem C que retorna o controle ao SO, passando um código de retorno e terminando a execução do programa.

fprintf()

A função **fprintf()** é usada para escrever dados formatados em um arquivo em C, assim como a função printf() é usada para escrever dados formatados na saída padrão (normalmente o console). A sintaxe básica da função fprintf() é a seguinte:

```
int fprintf (FILE *stream, const char *format, ...);
```

Onde:

- **stream**: O ponteiro para o arquivo no qual os dados serão gravados.
- **format**: string de formato que especifica como os dados devem ser formatados e escritos no arquivo.
- **...**: argumentos variáveis relativos aos valores que serão escritos no arquivo.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main() {

    FILE *arquivo;
    int numero = 125;
    char palavra[] = "Esta é uma frase que será gravada no arquivo!";

    arquivo = fopen("saida.txt", "w");

    if (arquivo == NULL) {
        printf("Erro ao abrir o arquivo.\n");
        exit(1);
    }

    fprintf(arquivo, "Número: %d\nPalavra: %s\n", numero, palavra);

    fclose(arquivo);
    return 0;
}
```

fscanf()

A função **fscanf()** é usada para ler dados formatados de um arquivo em C, assim como a função scanf() é usada para ler dados formatados da entrada padrão (normalmente o teclado). A sintaxe básica da função fscanf() é a seguinte:

```
int fscanf (FILE *stream, const char *format, ...);
```

Onde:

- **stream**: O ponteiro para o arquivo que contém os dados a serem lidos.
- **format**: string de formato que especifica como os dados devem ser lidos do arquivo.
- **...:** argumentos variáveis relativos aos valores que serão escritos no arquivo.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main() {
    FILE *arquivo;
    int numero;
    char palavra[50];

    arquivo = fopen("arquivo.txt", "r");
    if (arquivo == NULL) {
        printf("Erro ao abrir o arquivo.\n");
        exit(1);
    }

    // Lendo um número inteiro e uma palavra do arquivo

    fscanf(arquivo, "%d %s", &numero, palavra);
    printf("Número: %d\nPalavra: %s\n", numero, palavra);
    fclose(arquivo);
    return 0;
}
```

Edite o arquivo texto colocando o número depois da string e observe o resultado da execução e reflita sobre que conclusões podem ser obtidas.

fgets()

A função **fgets()** em C é usada para ler uma linha de texto de um arquivo ou da entrada padrão (normalmente o teclado) e armazená-la em uma string. Ela é útil para ler linhas completas de texto, incluindo espaços em branco, e armazená-las em variáveis de string. A sintaxe é detalhada a seguir:

```
char *fgets(char *str, int num, FILE *stream);
```

Onde:

- **stream**: ponteiro para o arquivo no qual os dados serão gravados.
- **str**: ponteiro para a string na qual os dados lidos serão armazenados.
- **num:** múmero máximo de caracteres a serem lidos, incluindo os caracteres '\n' e '\0'.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main() {
    FILE *arquivo;
    char linha[100];

    arquivo = fopen("dados.txt", "r");
    if (arquivo == NULL) {
        printf("Erro ao abrir o arquivo.\n");
        exit(1);
    }

    // Lê e exibe cada linha do arquivo

    while (fgets(linha, sizeof(linha), arquivo) != NULL) {
        printf("%s", linha);
    }

    fclose(arquivo);
    return 0;
}
```

fputs()

A função **fputs()** é usada para escrever uma string em um arquivo. Ela é útil para gravar uma sequência de caracteres em um arquivo, sem a formatação adicional que as funções de formatação. A função fputs() não adiciona automaticamente caracteres de nova linha e tem a seguinte sintaxe:

```
int fputs(const char *str, FILE *stream);
```

Onde:

- **stream**: ponteiro para o arquivo no qual os dados serão gravados.
- **str**: ponteiro para a string na qual os dados lidos serão armazenados.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main() {

    FILE *arquivo;
    arquivo = fopen("saida.txt", "w");
    if (arquivo == NULL) {
        printf("Erro ao abrir o arquivo.\n");
        return 1;
    }

    // Escrevendo uma string no arquivo

    const char *mensagem = "Esta é uma mensagem de exemplo.";
    fputs(mensagem, arquivo);

    fclose(arquivo);
    return 0;
}
```

Q02 Descreva o funcionamento do seguinte código-fonte:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main() {

    FILE *f1, *f2;

    f1 = fopen("entrada.txt", "r");
    f2 = fopen("saida.bin", "w");

    fclose(f1);
    fclose(f2);

    return 0;
}
```

Q03 Descreva o funcionamento do seguinte código-fonte:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main() {

    FILE *f1, *f2;
    int x;
    f1 = fopen("entrada.txt", "r");
    f2 = fopen("saida.bin", "w");

    fscanf(f1, "%d", &x);
    fprintf(f2, "%d", x);

    fclose(f1);
    fclose(f2);
    return 0;
}
```

Q04

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main() {

    FILE *arquivo;
    int X;
    arquivo = fopen("entrada.txt", "r");
    if (arquivo == NULL) {
        printf("Erro ao abrir o arquivo.\n");
        exit(1);
    }

    while (fscanf(arquivo, "%d", &X) != EOF) {
        printf("%d\n", X);
    }

    fclose(arquivo);
    return 0;
}
```

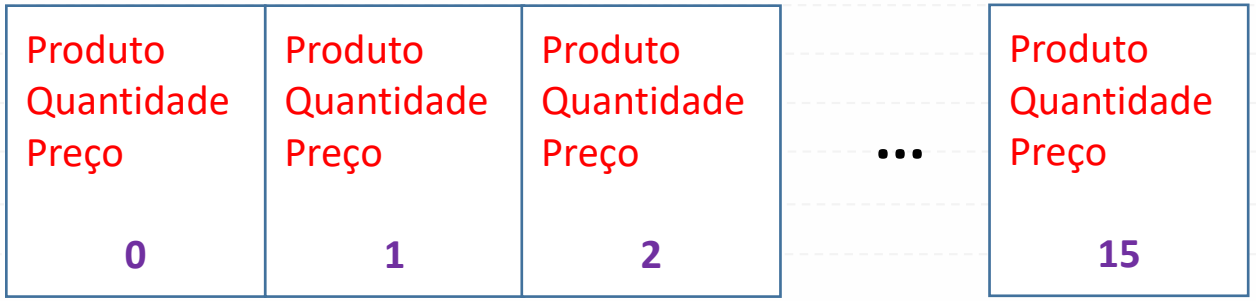
Q05 Escreva um programa que solicite do usuário os nomes de 2 arquivos texto para, a seguir, criar um terceiro arquivo, também texto, com o conteúdo dos dois arquivos fornecidos (o conteúdo do primeiro arquivo seguido do conteúdo do segundo arquivo).

Q06

Escreva um programa que leia um arquivo texto contendo uma lista de compras em que cada linha possui o nome do produto, a quantidade e o valor unitário de cada item para, em seguida, informar o valor total da compra.

Caso o arquivo texto tenha:

Arroz 2 5.50
Feijão 3 3.00
Carne 1 15.75
Leite 4 2.50



A saída (na tela) deverá ser:

1) Arroz: 2 * 5.50 = 11.00
2) Feijão: 3 * 3.00 = 9.00
3) Carne: 1 * 15.75 = 15.75
4) Leite: 4 * 2.50 = 10.00

O valor total da compra é R$ 45.75

Q07

Escreva um programa que leia, do dispositivo de entrada padrão, os nomes de dois arquivos e imprima mensagem indicando se os conteúdos dos dois arquivos, que você pode considerar que são arquivos-texto, são iguais entre si ou não. Além disso, se não forem iguais, a mensagem deve conter, além da informação de que são diferentes, qual é a posição do primeiro caractere distinto encontrado, e quais são estes caracteres distintos, nos dois arquivos. A posição i é a do i-ésimo caractere (posições começam de 1). O programa deve testar se os arquivos existem e emitir mensagem apropriada, no dispositivo padrão de erro, caso um deles ou ambos não existam.

Q08

Faça um programa que crie um arquivo TEXTO em disco, com o nome “dados.txt”, e escreva neste arquivo uma contagem que vá de 1 até 100, com um número em cada linha.

Q09

Faça um programa que leia (do teclado) um cadastro de 10 alunos, indicando o nome, nota1, nota2; calcule a nota média de cada aluno e depois escreva em um arquivo texto os dados de cada aluno: nome, nota1, nota2 e média.

Observação:

As notas e média deverão ser apresentadas como valores que possuem até 2 casas após a vírgula.

Q10

Faça um que abra um arquivo HTML e elimine todas as “tags” do texto, ou seja, o programa deve gerar um novo arquivo em disco que elimine todas as tags HTML, que são caracterizadas por comandos entre “<>”. Veja abaixo um exemplo de um texto em HTML e do texto que deverá ser gerado pelo programa após eliminar as tags Html

```
<HTML>
<HEAD>
<TITLE>Minha Pagina Web</TITLE>
</HEAD>
<BODY>
Meu Texto<BR>
Minha Imagem<IMG SRC=”img.jpg”>
<A HREF=”pag1.html”>Meu Link</A>
</BODY>
</HTML>
```

```
Minha Pagina Web


Meu Texto
Minha Imagem
Meu Link
```